## AMENDMENTS TO THE CLAIMS

This listing of claims replaces all prior versions and listings, of claims in the application:

Listing of Claims

1. (Currently Amended) ~~Code~~ A code generator ~~(60)~~ for generating an orthogonal code having a spreading ~~factor SF~~ factor (SF) and an ~~index k~~ index (k), wherein the spreading ~~factor SF~~ factor (SF) is selectable from values in a range $1 < SF \le SF_{max}$ with $SF_{max}$ denoting a maximum spreading factor, said code generator ~~(60)~~ ~~including~~ comprising:

[[a)]] an index conversion unit ~~(61)~~ for converting ~~said index k~~ the index (k) into a modified ~~index j~~ index (j) associated with a corresponding code having the maximum spreading factor[[,]]; and

[[b)]] a logic unit ~~(62)~~ for performing logic operations on bits of ~~said~~ the modified ~~index j~~ index (j) and bits of a counter ~~value i~~ value (i), thereby generating a code bit of ~~said~~ the orthogonal code.

2. (Currently Amended) ~~Code~~ The code generator according to claim 1, wherein said corresponding code is one of: an ~~OVSF~~ orthogonal variable spreading factor (OVSF) code, a Hadamard code, and a Walsh code.

3. (Currently Amended) ~~Code~~ The code generator according to claim 1 ~~or 2~~, wherein said index conversion unit ~~(61)~~ includes multiplication means ~~(71, 72)~~ for multiplying ~~said index k~~ the index (k) with a value of $SF_{max}/SF$.

4. (Currently Amended) ~~Code~~ The code generator according to claim 3, wherein said multiplication means ~~(71, 72)~~ includes:

[[-]] a mapping unit ~~(72)~~ for mapping ~~said~~ the spreading ~~factor SF~~ factor (SF) to a ~~number s~~ number (s) equal to $\log_2\{SF_{max}/SF\}$,

[[-]] a shift register ~~(71)~~ adapted to receive and store ~~said index k~~ the index (k) in binary representation, further adapted to receive ~~said number s~~ the number (s) and to shift the stored ~~index k by s~~ index (k) by (s) bit positions in the direction of more significant bit positions.

5. (Currently Amended) ~~Code~~ The code generator according to ~~one of the preceding claims~~ claim 1, wherein ~~said~~ the index conversion unit ~~(61)~~ includes a permutation unit ~~(73)~~ for permuting the bits of ~~said index k~~ the index (k).

6. (Currently Amended) ~~Code~~ The code generator according to claim 3 ~~or 4~~, wherein said index conversion unit ~~(61)~~ includes:

[[-]] a permutation unit ~~(73)~~ for permuting the bits of ~~said index k,~~ the index (k); and

[[- a]] selection means ~~(74)~~ for selecting, ~~in dependence of~~ depending upon a mode signal indicating a desired type of said orthogonal code, the output of ~~said~~ the permutation unit ~~(73)~~ or the output of ~~said~~ the shift register ~~(71)~~, thereby generating ~~said~~ the modified ~~index j~~ index (j).

7. (Currently Amended) ~~Code~~ The code generator according to ~~one of the preceding claims~~ claim 1, wherein said logic unit ~~(62)~~ includes:

[[-]] adding means ~~(81-1, ..., 81-9)~~ for performing binary AND operations, wherein ~~each~~ the adding means is adapted to receive a bit of ~~said~~ the modified ~~index j~~ index (j) and a bit of ~~said~~ the counter ~~value i~~ value (i), and is further adapted to output a binary output value representing a binary AND combination of ~~said~~ the two bits[[,]]; and

[[-]] combining means ~~(82)~~ for combining ~~said~~ the binary output values into ~~said~~ the code bit.

8. (Currently Amended) ~~Code~~ The code generator according to claim 7, wherein said combining means ~~(82)~~ includes means for performing binary XOR operations ~~(82-1, ..., 82-8)~~.

9. (Currently Amended) ~~Code~~ The code generator according to ~~one of the preceeding claims~~ claim 1, further ~~including~~ comprising a counter ~~(63)~~ for generating ~~said~~ the counter ~~value i~~ value (i).

10. (Currently Amended) ~~Parallel~~ A parallel code generator ~~(90)~~ for concurrently generating a number $p > 1$ orthogonal codes having respective spreading factors ~~(~~$SF_1$, ..., $SF_p$~~)~~ and indices ~~(~~$k_1$,..., $k_p$~~)~~, wherein the spreading factors are selectable from values in a range $1 < SF_1, ..., SF_p \leq SF_{max}$~~.~~, with $SF_{max}$ denoting a maximum spreading factor, said parallel code generator ~~(90) including~~ comprising:

[[a)  p]] a number (p) of code generators ~~(90-1, 90-2, ..., 90-p) according to one of the claims 1 to 8~~, each for generating one of ~~said~~ the p orthogonal codes having a particular one of ~~said~~ the spreading factors and a particular one of ~~said~~ the indices, each of said (p) code generators including:

an index conversion unit for converting the index (k) into a modified index (j) associated with a corresponding code having the maximum spreading factor; and

a logic unit for performing logic operations on bits of the modified index (j) and bits of a counter value (i), thereby generating a code bit of the orthogonal code; and

[[b)]] a counter ~~(93)~~ for generating ~~said~~ the counter ~~value i~~ value (i) to be used by ~~said p~~ the (p) code generators.

11. (Currently Amended) ~~Parallel~~ A parallel code generator for concurrently generating a number $p > 1$ orthogonal codes having respective spreading factors ~~(~~$SF_1$, ..., $SF_p$~~)~~ and indices ~~(~~$k_1$,..., $k_p$~~)~~, wherein the spreading factors are selectable from values in a range $1 < SF_1, ..., SF_p \leq SF_{max}$~~.~~, with $SF_{max}$ denoting a maximum spreading factor, said parallel code generator ~~including~~ comprising:

[[p]] a number (p) of code generators ~~according to claim 9~~, each of said code generators including:

an index conversion unit for converting the index (k) into a modified index (j) associated with a corresponding code having the maximum spreading factor;

a logic unit for performing logic operations on bits of the modified index (j) and bits of a counter value (i), thereby generating a code bit of the orthogonal code; and a counter for generating the counter value (i);

wherein each ~~for generating~~ of the code generators generates one of ~~said p~~ the (p) orthogonal codes having a particular one of ~~said~~ the spreading factors and a particular one of ~~said~~ the indices.

12. (Currently Amended) ~~Code generation method for~~ A method of generating an orthogonal code having a spreading ~~factor SF~~ factor (SF) and an ~~index k~~ index (k), wherein the spreading ~~factor SF~~ factor (SF) is selectable from values in a range $1 < SF \leq SF_{max}$, with $SF_{max}$ denoting a maximum spreading factor, said ~~code generation~~ method ~~including~~ comprising the steps of:

a) converting ~~(101) said index k~~ the index (k) into a modified ~~index j~~ index (j) associated with a corresponding code having the maximum spreading factor[[,]];

b) ~~initialising (102)~~ initializing a counter ~~value i,~~ value (i);

c) performing logic operations ~~(103)~~ on bits of ~~said~~ the modified ~~index j~~ index (j) and bits of ~~said~~ the counter ~~value i~~ value (i), thereby generating a code bit of ~~said~~ the orthogonal code[[,]];

d) incrementing ~~(104) said~~ the counter ~~value i~~ value (i) by one[[,]]; and

e) repeating ~~said step of performing logic operations (103) and said step of incrementing(104)~~ steps c) and d) until a desired number of code bits has been generated.

13. (Currently Amended) ~~Code generation~~ The method according to claim 12, wherein said corresponding code is one of: an ~~OVSF~~ orthogonal variable spreading factor (OVSF) code, a Hadamard code, and a Walsh code.

14. (Currently Amended) ~~Code generation~~ The method according to claim 12 ~~or 13~~, wherein ~~said step of converting (101)~~ step a) includes ~~a step of~~ multiplying ~~(111, 112, 113) said index k~~ the index (k) with a value of $SF_{max}/SF$.

15. (Currently Amended) ~~Code generation~~ The method according to claim 14, wherein said step of multiplying ~~(111, 112, 113)~~ includes the steps of:

[[-]] mapping ~~(111) said~~ the spreading ~~factor SF~~ factor (SF) to a ~~number s~~ number (s) equal to $\log_2\{SF_{max}/SF\}$[[,]];

[[-]] storing ~~(112) said index k~~ the index (k) in binary representation in a shift register[[,]]; and

[[-]] shifting ~~(113)~~ the stored ~~index k by s~~ index (k) by (s) bit positions in the direction of more significant bit positions.

16. (Currently Amended) ~~Code generation~~ The method according to ~~one of the claims 12 to 15~~ claim 12, wherein ~~said step of converting (101)~~ step a) includes [[a step of]] permuting ~~(114)~~ the bits of ~~said index k~~ the index (k).

17. (Currently Amended) ~~Code generation~~ The method according to claim 14 ~~or 15~~, wherein ~~said step of converting (101)~~ step a) includes the steps of:

[[-]] permuting ~~(114)~~ the bits of ~~said index k,~~ the index (k); and

[[-]] selecting ~~(115)~~, ~~in dependence of~~ depending upon a mode signal indicating a desired type of ~~said~~ the orthogonal code, the permuted index or the shifted index, thereby generating ~~said~~ the modified ~~index j~~ index (j).

18. (Currently Amended) ~~Code generation~~ The method according to ~~one of the claims 12 to 17~~ claim 12, wherein ~~said step of performing logic operations (103)~~ step c) includes the steps of:

[[-]] performing binary AND operations ~~(121)~~, wherein each operation is adapted to combine a bit of ~~said~~ the modified ~~index j~~ index (j) and a bit of ~~said~~ the counter ~~value i~~ value (i), and to output a binary output value representing a binary AND combination of ~~said~~ the two bits[[,]]; and

[[-]] combining ~~(122) said~~ the binary output values into ~~said~~ the code bit.

19. (Currently Amended) ~~Code generation~~ The method according to claim 18, wherein said step of combining ~~(122)~~ includes ~~a step of~~ performing binary XOR operations.

20. (Canceled)

21. (New) A computer program product directly loadable into an internal memory of a communication unit, said product comprising software code portions that generate an orthogonal code having a spreading factor (SF) and an index (k), wherein the spreading factor (SF) is selectable from values in a range $1 < SF \leq SF_{max}$, with $SF_{max}$ denoting a maximum spreading factor, wherein, when the product is run on a processor of the communication unit, the following steps are performed:

a) converting the index (k) into a modified index (j) associated with a corresponding code having the maximum spreading factor;

b) initializing a counter value (i);

c) performing logic operations on bits of the modified index (j) and bits of the counter value (i), thereby generating a code bit of the orthogonal code;

d) incrementing the counter value (i) by one; and

e) repeating steps c) and d) until a desired number of code bits has been generated.